2005/03/23 Motion Estimation by Chu, Lin, and Hsien
Members:
D91922016 朱威達
R93922010 林聖凱
R93922044 謝俊瑋

Motion Estimation

There are three main types (or applications) of motion estimation:

- Parametric motion (image alignment)

The main idea of parametric motion is to combine two or more picture of the same place with different small sections (such as Fig.1 and Fig.2), and combine them in to a whole picture of the place (such as Fig.3). In order to combine these photos, we have to find the motion of the camera, and make the view point of these photos the same, and then we can perfectly combine these photos.

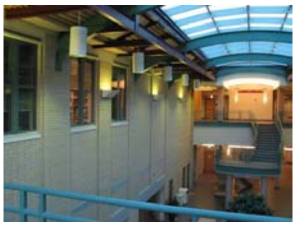
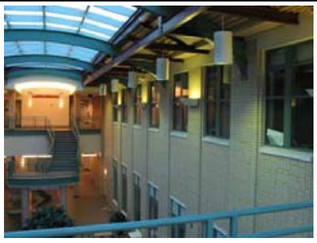
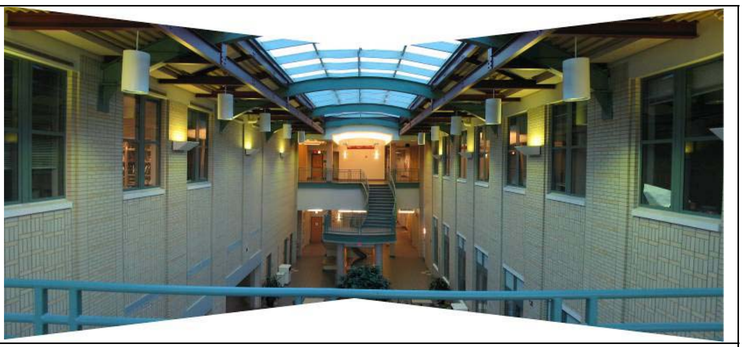
		

Fig.1	Fig.2	Fig.3

- Tracking

Tracking is to track the motion of specified point from one picture to another. Usually these points will have special characteristics that allow us to have higher accuracy of their motions.

- Optical flow

Optical flow is to determine the motion of every point from one photo to another, and these motions must have consistency. Therefore their accuracy might be worse than tracking.

For the motion estimation, there exist tree assumptions that can help us solve this problem:

- Brightness consistency

Image measurements (e.g. brightness) in a small region remain the same although their location may change.

- Spatial coherence

Neighboring points in the scene typically belong to the same surface and hence typically have similar motions. Since they also project to nearby points in the image, we expect spatial coherence in image flow.

- Temporal persistence

The image motion of a surface patch changes gradually over time.

Image registration:

Goal: register a template image J(x) and an input image I(x), where $x = (x, y)^T$

For 3 problems above, I and J are defined as followed:

Image alignment: I(x) and J(x) are two images

Tracking: J(x) is the image at time t, and I(x) is a small patch around the point p in the image at $t+1$.

Optical flow: J(x) and I(x) are image of t and $t+1$.

Simple approach:

We can solve this problem simply by minimizing the brightness difference that is use the difference of brightness as our measurement.

Simple SSD algorithm:

For each offset (u, v)

Compute $E(u,v)$;

Choose (u, v) which minimizes $E(u,v)$;

$$E(u,v) = \sum_{x,y} \left(I(x+u, y+v) - J(x,y) \right)^2$$

Problems:

- Not efficient, we have to calculate it pixel by pixel and repeat many times.
- No sub-pixel accuracy

Lucas-Kanade algorithm

This algorithm was proposed by B.D. Lucas and T. Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision, Proceedings of the 1981 DARPA Image Understanding Workshop, 1981, pp121-130.

First we consider Newton's method of finding root for f(x) = 0, by Taylor's expansion, the function f(x) can be approximated by $f(x+\varepsilon) = f(x_0) + f'(x_0)\varepsilon + \frac{1}{2} f''(x_0)\varepsilon^2 + \ldots$

So in each iteration, set the $\varepsilon_n = -\frac{f(x_n)}{f'(x_n)}$ and $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, we can the approximate the root of f(x)=0.

In the 2-dimension case we want to minimize $E(u,v) = \sum_{x,y} \left(I(x+u, y+v) - J(x,y) \right)^2$, and by using the same technique as Newton's method to find the root of

$0 = \frac{\partial E}{\partial u} = \sum_{x,y} 2I_x \left(I(x,y) - J(x,y) + uI_x + vI_y \right)$ and

$0 = \frac{\partial E}{\partial v} = \sum_{x,y} 2I_y \left(I(x,y) - J(x,y) + uI_x + vI_y \right)$, we can have Lucas-Kanade algorithm.

For parametric model, we need to transfer $E(u,v) = \sum_{x,y} \left(I(x+u, y+v) - J(x,y) \right)^2$ into

$$E(\mathbf{p}) = \sum_{\mathbf{x}} \left(I(\mathbf{W}(\mathbf{x};\mathbf{p})) - J(\mathbf{x}) \right)^2$$

And finally the algorithm becomes:

```
iterate
    warp I with W(x;p)
    compute error image J(x,y)-I(W(x,p))
    compute gradient image
    evaluate Jacobian        at (x;p)
    compute $\nabla I \frac{\partial W}{\partial p}$
    compute Hessian
    compute $\sum_x \left[ \nabla I \frac{\partial W}{\partial p} \right] [J(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$
    solve $\Delta p$
    update p by p+$\Delta p$
until converge
```

$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T [J(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

Tracking

- Goal of Tracking

Given two images, we'd like to know where the corresponding point of image1 is in the image2.

- brightness constancy:

The basic assumption behind SSD minimization is brightness constancy.

$$I(x+u, y+v, t+1) - I(x,y,t) = 0$$

At a point a small distance away, and a small time later, the intensity is

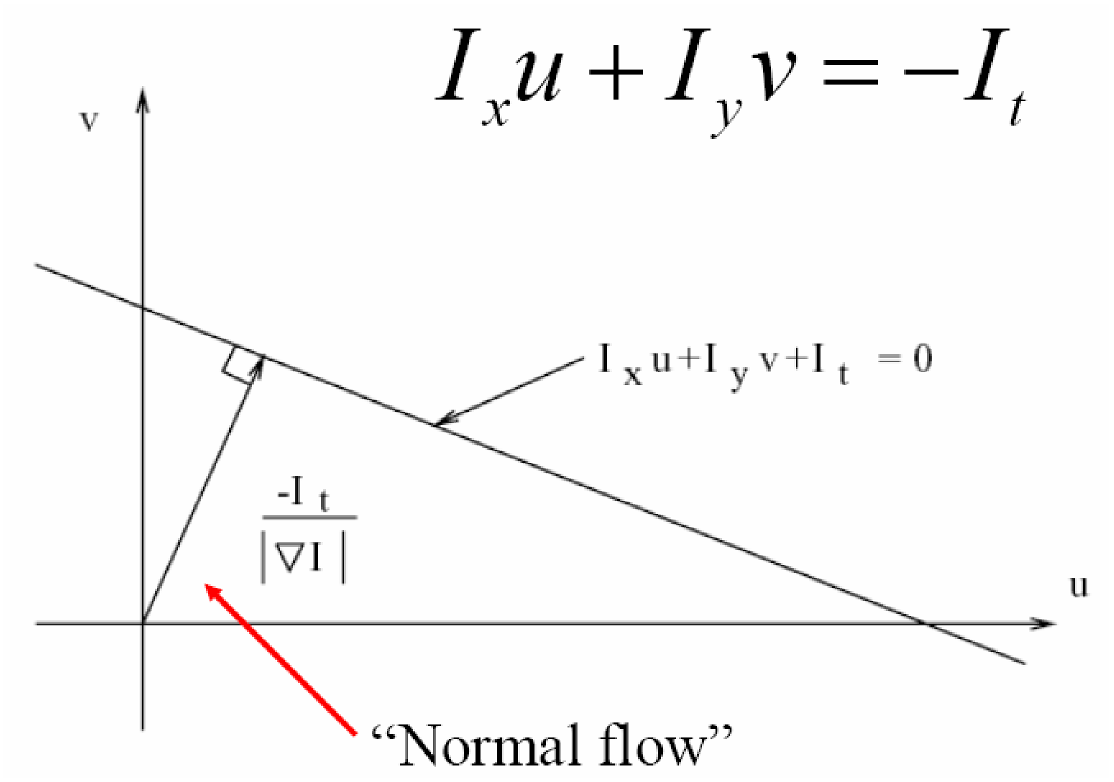
$$I(x,y,t) + uI_x(x,y,t) + vI_y(x,y,t) + I_t(x,y,t) - I(x,y,t) \approx 0$$

$$uI_x(x,y,t) + vI_y(x,y,t) + I_t(x,y,t) = 0$$

$$I_x u + I_y v + I_t = 0$$

The derived equation is also called “Optical flow constraint equation”.

If only one image pixel is used (one equation in two unknowns), the result will be a line.



Normal flow could be derived as following:

Normal speed: $S = \vec{v} \cdot \hat{n}$ (projection of velocity onto normal)

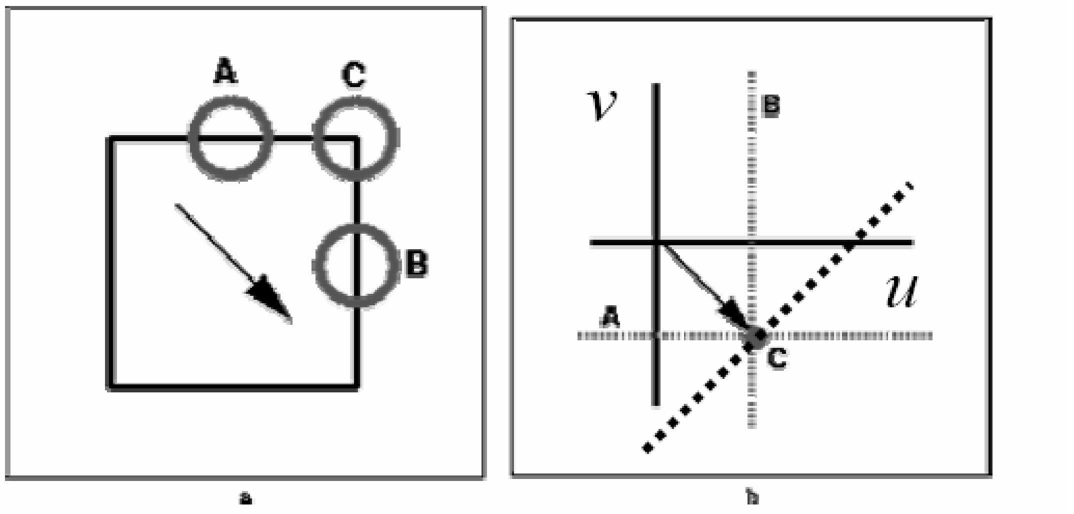
Normal Velocity: $\overline{v_\perp} = s \cdot \hat{n}$

Since $\vec{v} = \frac{-I_t}{\nabla I}$ and $\hat{n} = \frac{\nabla I}{\|\nabla I\|_2}$,

$$\overline{v_\perp} = \vec{v} \cdot \hat{n} \cdot \hat{n} = \frac{-I_t}{\nabla I} \cdot \frac{\nabla I}{\|\nabla I\|_2} \cdot \frac{\nabla I}{\|\nabla I\|_2}$$

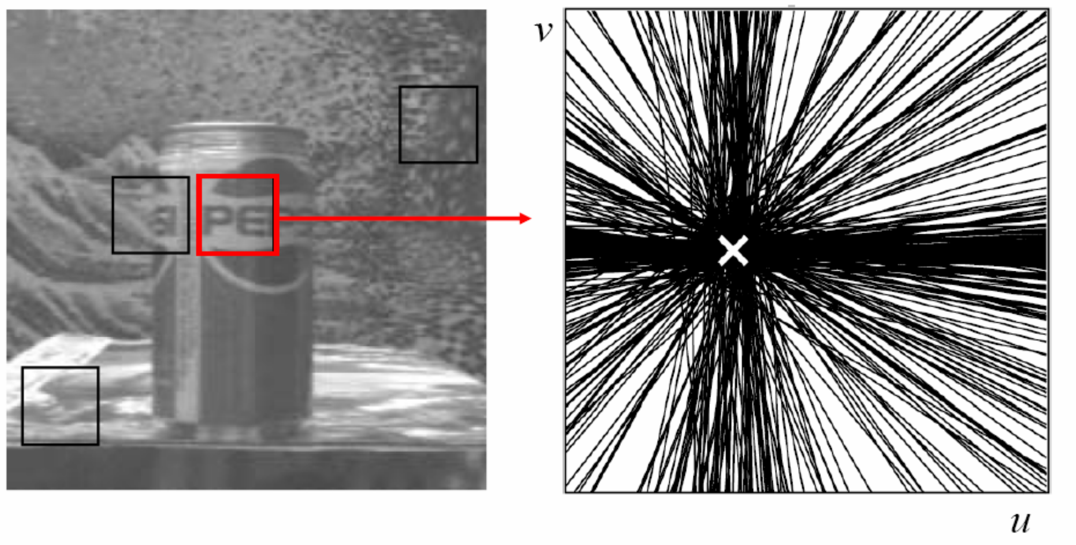
$$\overline{v_\perp} = \vec{v} \cdot \hat{n} \cdot \hat{n} = \frac{-I_t \cdot \nabla I}{\|\nabla I\|_2^2}$$

In order to get the accurate result, the multiple constraints are combined to get an estimate of the velocity.

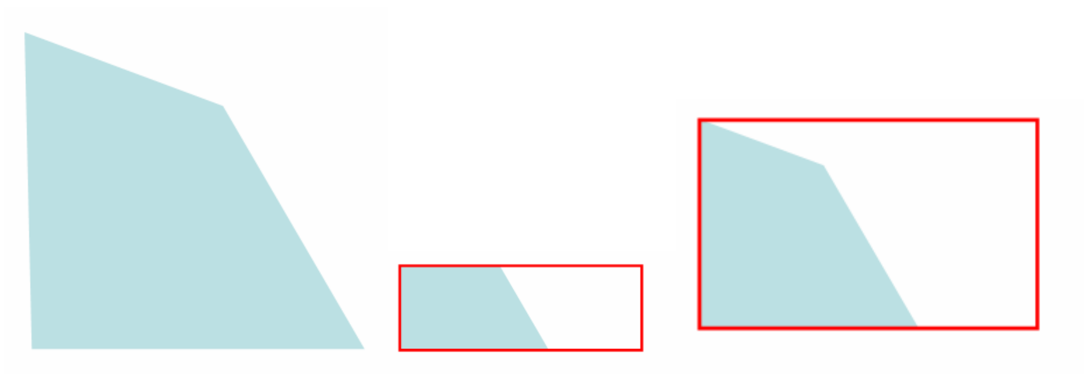


- Area-based method

Because of the disadvantage of the aforementioned methods, the area based method is used. Under the assumption of “Spatial smoothness”, we could use this technique to find out the best match. The method is similar to “multiple constraint”, and the best match can be easily found.



However, the size of block and the range of the block are important issues. Larger window reduces ambiguity, but easily violates spatial smoothness assumption. Beside where to search is also an important factor.



In the above graph, the middle image reveals insufficient information about the original image and hard to find out the best match. At the meanwhile, the rightmost image is good enough to find the best match.

In the remaining section, the math equations are derived.

Assume spatial smoothness

$$E(u,v)=\sum_{x,y}\left(I_x u + I_y v + I_t\right)^2$$

$$\frac{\partial E}{\partial u}=\sum_R (I_x u + I_y v + I_t) I_x = 0$$

$$\frac{\partial E}{\partial v}=\sum_R (I_x u + I_y v + I_t) I_y = 0$$

$$\left[\sum_R I_x^2\right]u+\left[\sum_R I_x I_y\right]v=-\sum_R I_x I_t$$

$$\left[\sum_R I_x I_y\right]u+\left[\sum_R I_y^2\right]v=-\sum_R I_y I_t$$

$$\begin{bmatrix}\sum I_x^2 & \sum I_x I_y\\ \sum I_y I_x & \sum I_y^2\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix}=\begin{bmatrix}-\sum I_x I_t\\ -\sum I_y I_t\end{bmatrix}$$ must be invertible

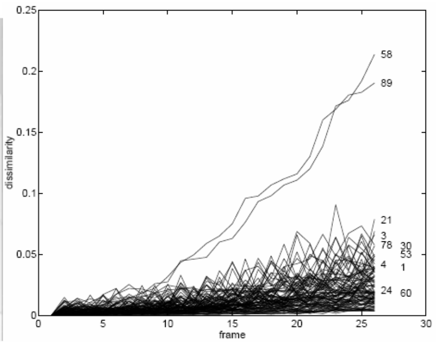
The eigenvalues tell us about the local image structure and how well we can estimate the flow in both directions.

- KLT tracking

The KLT tacking is selecting features by $min(\lambda_1,\lambda_2)>\lambda$ and the features are monitored by measuring dissimilarity. The details of KLT algorithm are not covered in the class. KLT is quite similar to SIFT but the motion is considered. However, in the practical use, SIFT can get the better result than KLT even if SIFT doesn't take motion into the consideration. There're two main reasons:

1. KLT accumulates errors.

2. KLT doesn't consider affine transformations.



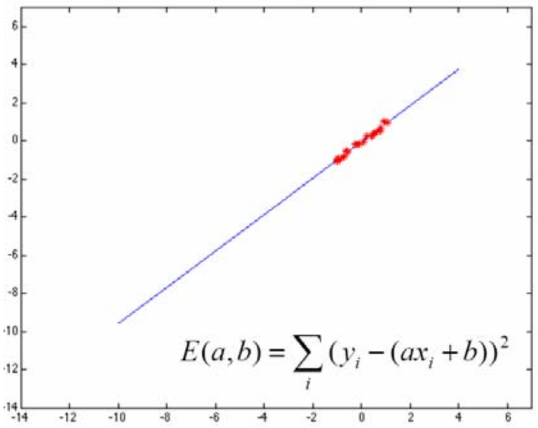
Optical Flow

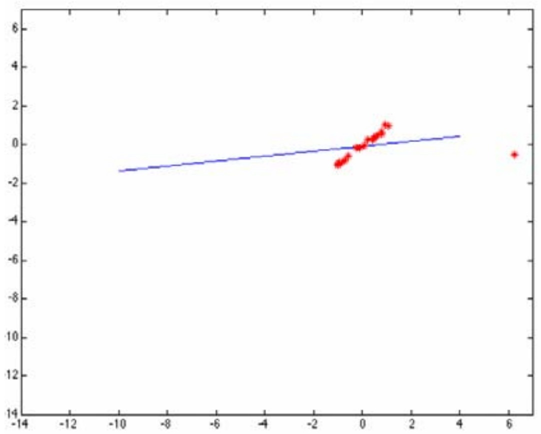
- Multiple motions via late many assumptions

Single motion assumption underlies the common correlation and gradient-based approaches and is violated when a region contains transparency, specular reflections, shadows, or fragmented occlusion. *Spatial coherence constraint* suffers from the same problem. It assumes that the flow within a neighborhood changes gradually since it is caused by a single motion.

- Least-square estimation is not robust

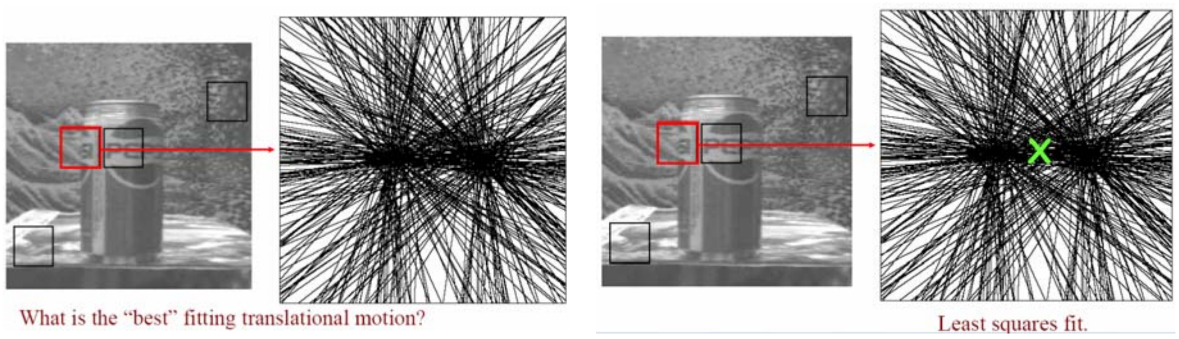
If two motions are present in a neighborhood, two sets of constraints will be consistent with different motions. When examining one of the motions, the constraints fro the other motion will appear as gross errors which we refer to as *outliers*. Unfortunately, least-square estimation is known to lack robustness in the presence of outliers. As shown in Figure 1, least-square estimation performs badly when some outliers exist. Figure 2 further shows the examples of using it to fit a result after compromising two different motions.





(a) (b)

Figure 1. (a) Use least-square to fit a line with coherent data. (b) Use least-square to fit a line some outliers.



(a) (b)

Figure 2. Least square estimation in multiple motions

- Robust statistical method

The main goals of robust statistics are to recover the structure that best fits the majority of the data while identifying and rejecting outliers or deviating substructures.

The problem of least-squares solution is that outlying measurements are assigned a high weight by a quadratic function, as shown in Figure 3. The influence of data points, which is proportional to the derivative of the quadratic function, increases linearly and without bound. Therefore, to reduce the influence of outliers, weighting function like truncated quadratic or other variations (Figure 4) could be used.

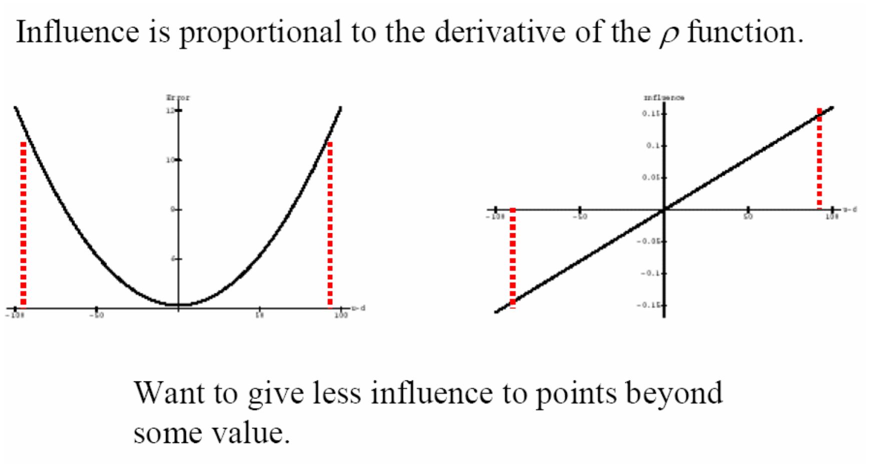
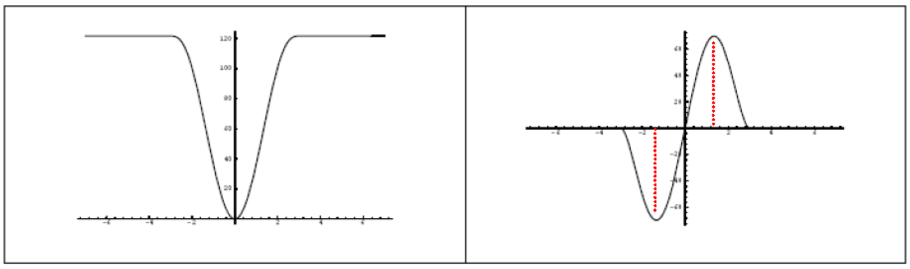


Figure 3. Weight and influence functions in least-square estimation.



Tukey's biweight.

Beyond a point, the influence begins to decrease.

Beyond where the second derivative is zero – outlier points

Figure 4. An example of weighting function to reduce the influence of outliers.

- Sample result of robust statistical method

In the sequence of Figure 5, the camera is stationary and a person is moving behind a plant resulting in fragmented occlusion. The robust estimation framework allows the dominant motion (of the plant and the background) to be accurately estimated while ignoring the motion of the person. Figures 6 and 7 show the sample results of detecting the major and secondary motions in this image sequence.

Figure 5. Sample image sequence.

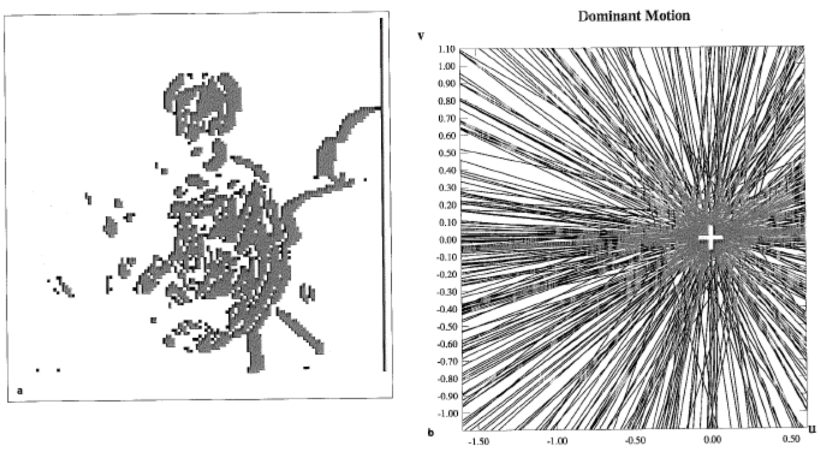


Figure 6. Robustly determining the dominant motion. (a) Violations of the single motion assumption shown as dark regions. (b) Constraint lines corresponding to the dominant motion.

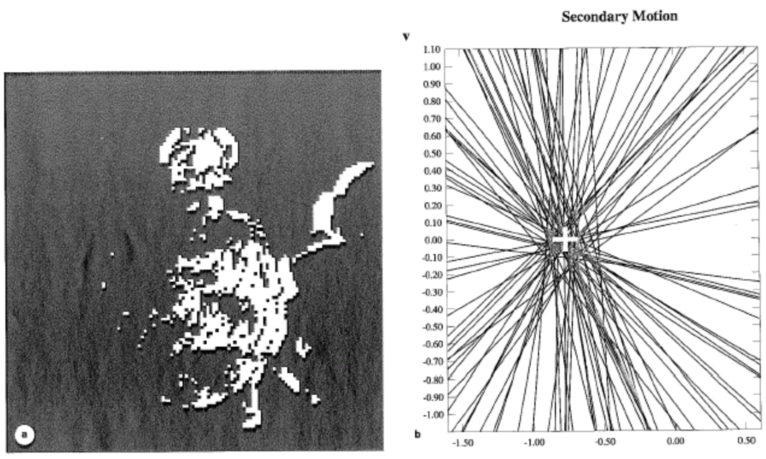


Figure 7. Estimating multiple motions. (a) White regions correspond to a second consistent motion (the person and his shadow). (b) Constraint lines corresponding to the secondary motion.

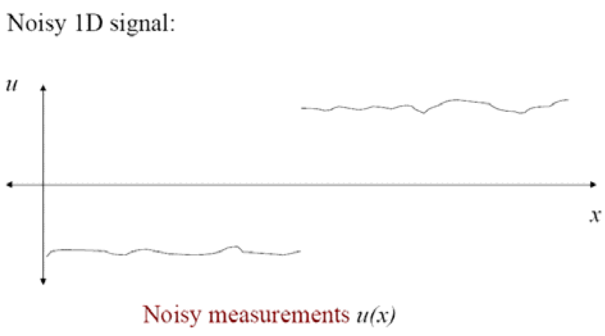
- Regularization and dense optical flow

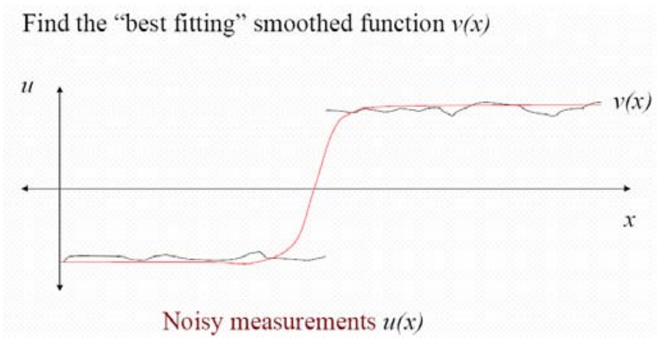
Assumption 1: *Neighboring points* in the scene typically belong to the same surface and hence typically *have similar motions*.

Assumption 2: Since they also project to nearby points in the image, we expect *spatial coherence* in image flow.

- Regularization

First, we want to formulate the regularization problem. Given a 1-D signal, as shown in Figure 8(a), we want to find a smoothed function that best fits this signal (Figure 8(b)). We both consider the data conservation and spatial smoothness. The regularization term is written as the formula in Figure 8(c), where λ controls the relative importance of the data conservation and spatial coherence terms. Further, we replace the least-square form with a robust function ρ to perform robust regularization (Figure 8(d)). That's how we robustly deal with motion boundary problem.





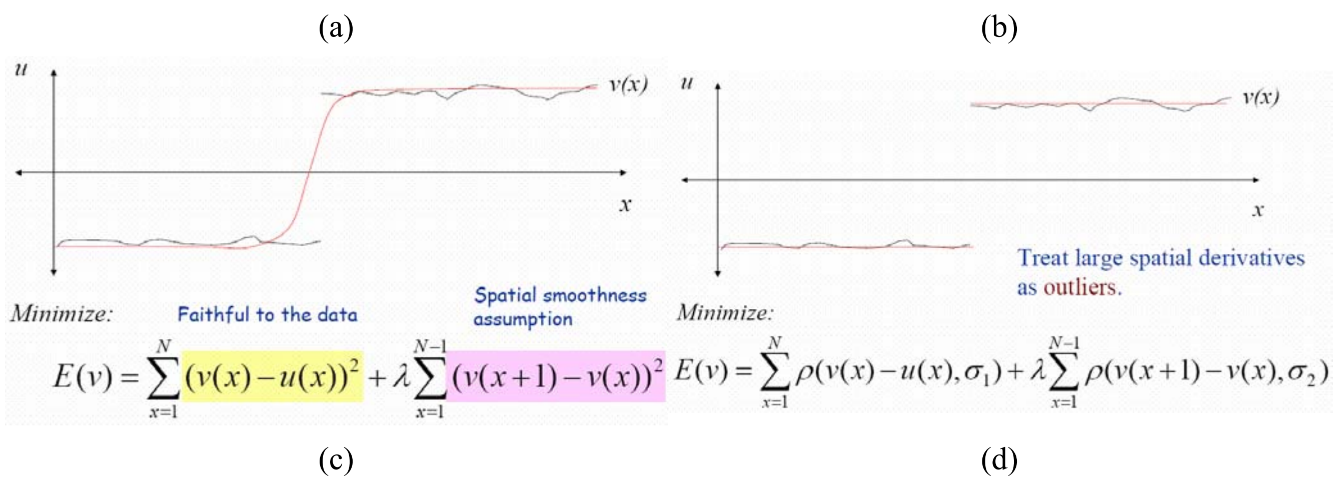


Figure 8. Formulation of regularization.

- Problems posted by motion boundaries

Figure 9 illustrates the situation which occurs at a motion boundary. With the least-square formulation the local flow vector $u_{i,j}$ is forced to be close to the average of its neighbors. When a motion discontinuity is present this results in smoothing across the boundary which reduces the accuracy of the flow field and obscures important structural information about the presence of an object boundary.

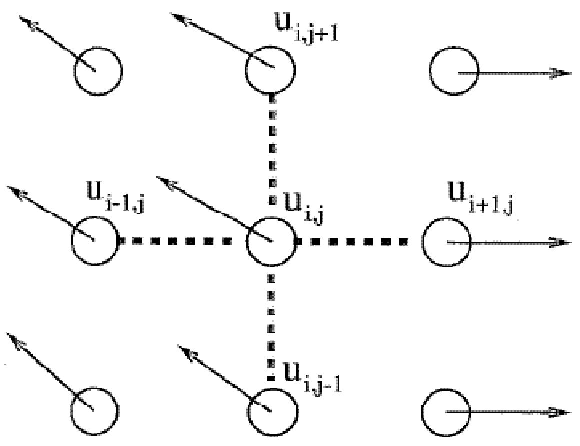
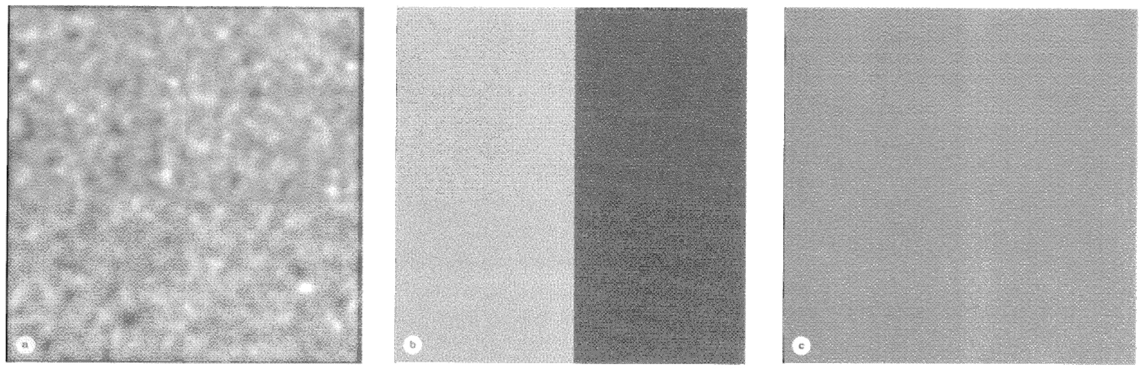


Figure 9. Smoothing across a flow discontinuity.

- An example of oversmoothing

Figure 10(a) shows a synthetic image sequence in which the left half of the image is stationary and the right half is moving one pixel to the left between frames. The horizontal and vertical components of the flow are shown with the magnitude of the flow represented by intensity, where black indicates motion to the left and up and gray indicates no motion. The true horizontal and vertical motions are shown in Figures 10(b) and (c), respectively.



(a) (b) (c)

Figure 10. Random noise example. (a) First random noise image in the sequence. (b) True horizontal motion (black = -1 pixel, white = 1 pixel, gray = 0 pixels). (c) True vertical motion.

Figure 11 shows a plot of the horizontal motion where the height of the plot corresponds to the recovered image velocity. Figure 11(a) shows that the application of the least-squares formulation results in motion estimates which very smoothly thus obscuring the motion boundary. When we recast the problem in the robust estimation framework, the problems of oversmoothing are reduced (Figure 11(b)).

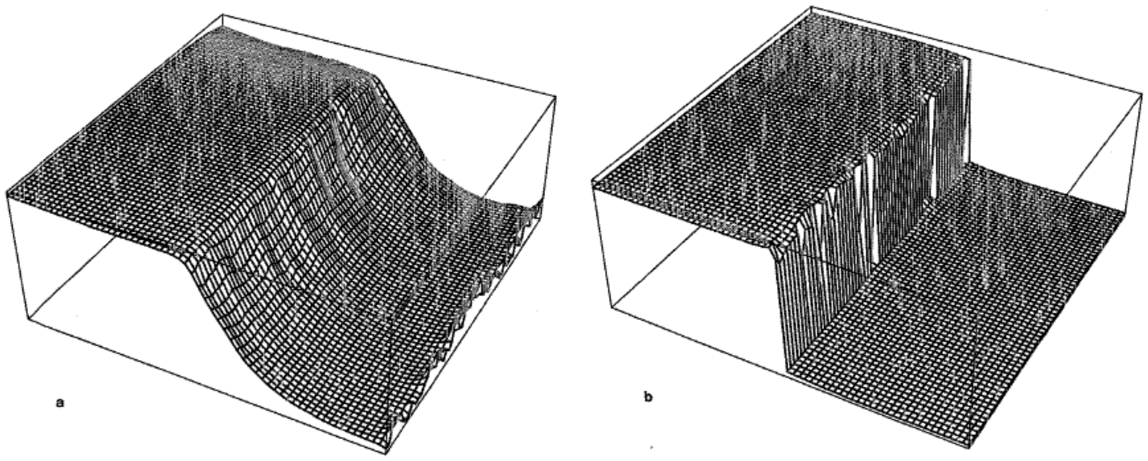


Figure 11. Horizontal displacement. The horizontal component of motion is interpreted as height and plotted. Plotting the results illustrates the oversmoothing of the least-square solution (a), and the sharp discontinuity which is preserved by robust estimation (b).

In another kind of presentation, we can see the estimation differences between quadratic and robust methods in Figure 12. The quadratic method introduces blur effects in motion boundaries, while robust method keeps sharp estimation in boundaries.

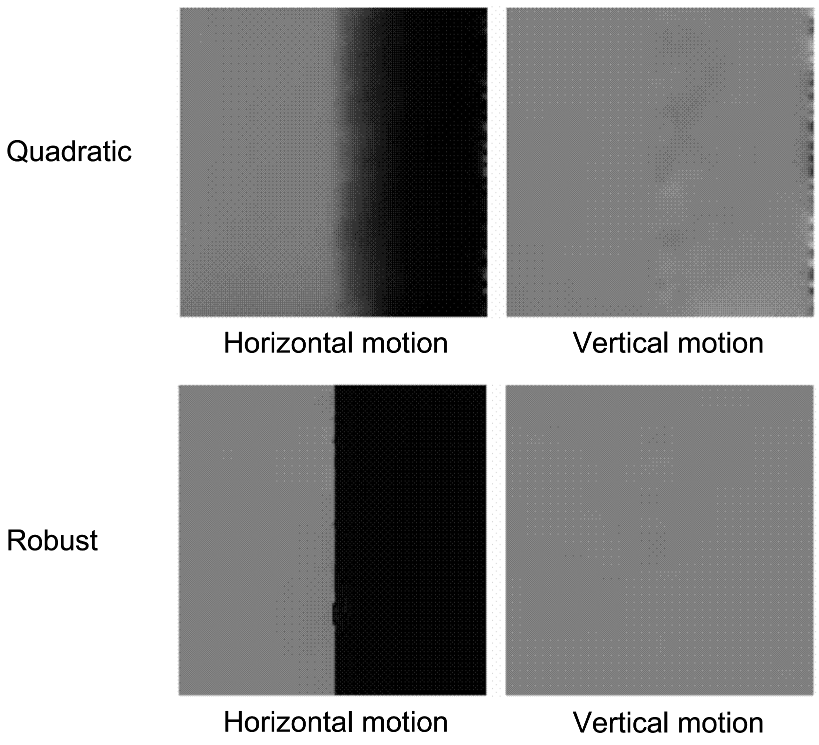


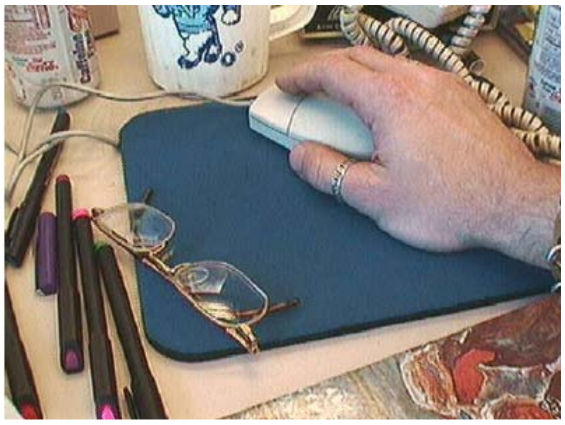
Figure 12. Estimation differences between the quadratic and robust methods.

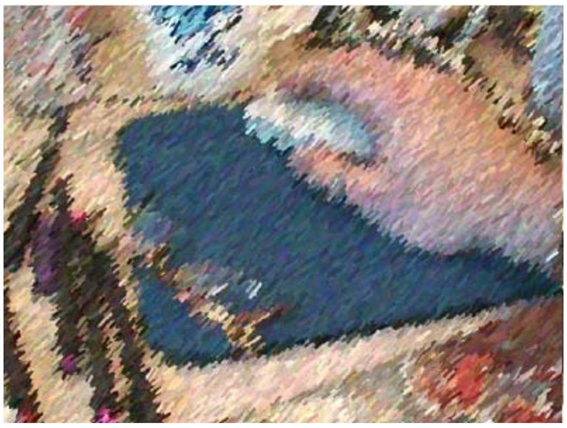
- Applications of optical flow – impressionist effects

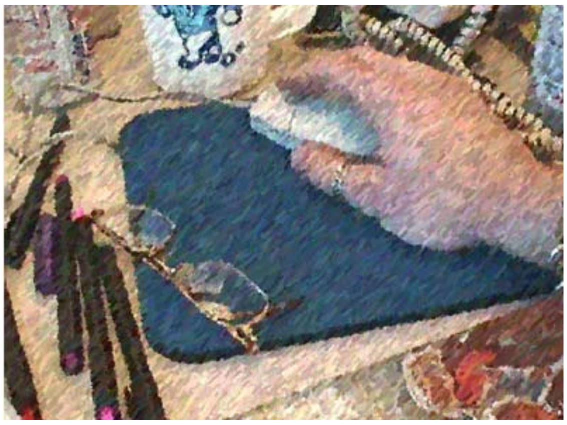
One application of optical flow is to produce impressionist effect for images or videos. Figure 13(b) is obtained after generating brush strokes which orientation is 45°. Brush generation method is modified so that brush strokes are clipped to edges in Figure 13(c). Next, an artist may not want to have all strokes drawn in the same direction. Therefore, the effect is improved that the stroke direction is decided by the gradient (Figure 13(d)). Furthermore, techniques for Figure 13(d) is modified such that with vanishing gradient magnitude are interpolated from surrounding regions (Figure 13(e)). Brush strokes may also be rendered with textured brush images that have r,g,b and alpha components. Figure 13(f) shows an example of applying texture brush. Details of this application please see "Peter Litwinowicz, Processing Images and Video for An Impressionist Effects, SIGGRAPH 1997."

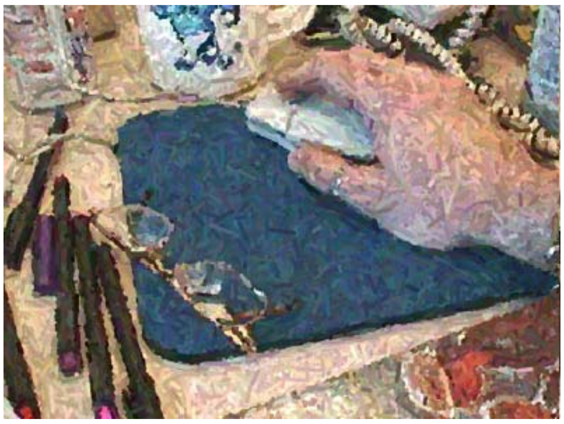
- Temporal coherence

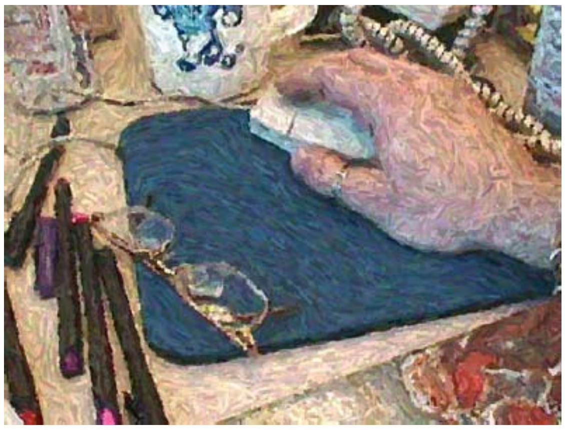
In a video sequence, the optical flow vector field is used as a displacement field to move the brush strokes to new locations in a subsequent frame. We must make sure to generate new strokes near the image boundaries to maintain temporal coherence. As shown in Figure 14, new brushes will be generated when regions are too sparse. Details of this application please see "Peter Litwinowicz, Processing Images and Video for An Impressionist Effects, SIGGRAPH 1997."


(a) Input image


(b) Brush


(c) Edge clipping


(d) Gradient


(e) Smooth gradient

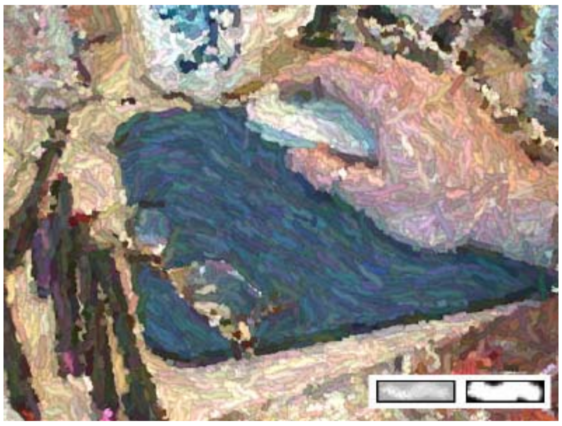

(f) Texture brush

Figure 13. Process of producing impressionist effect for images.

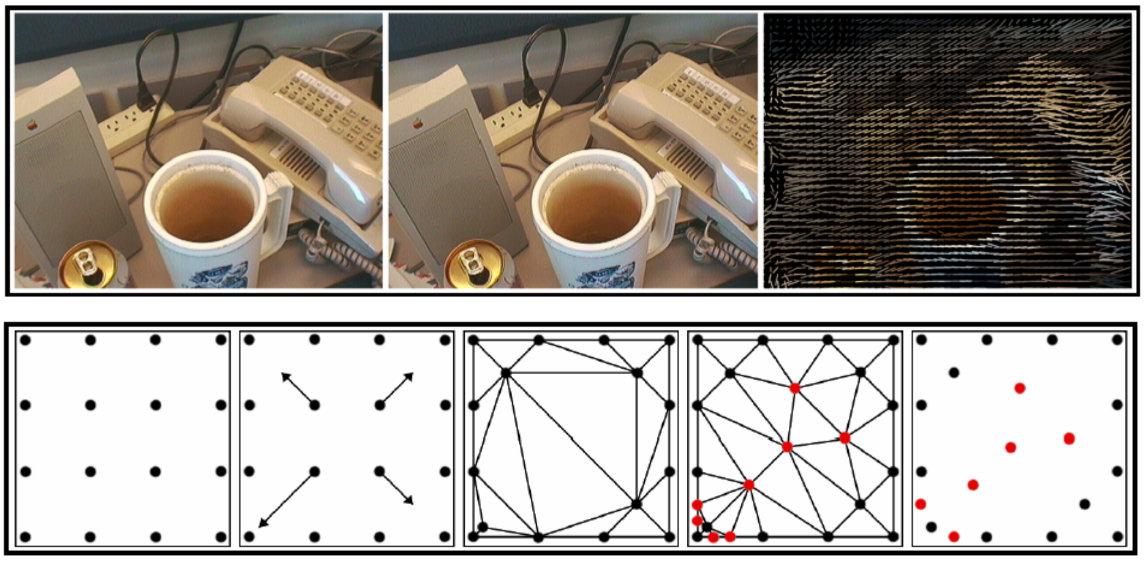


Figure 14. The process of generating new brushes.